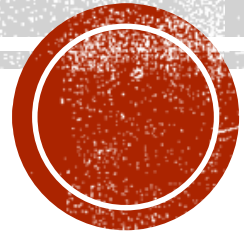
Yii2: Key Concepts

Oleh: Ahmad Syauqi Ahsan



Components

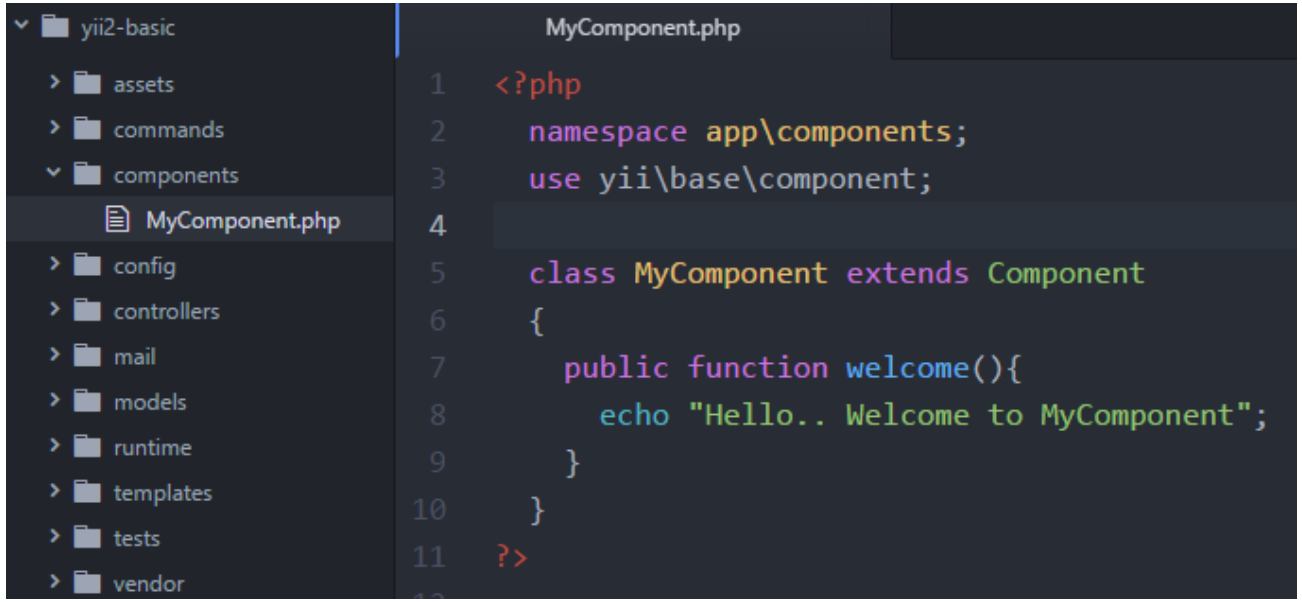
```
use yii\jui\DatePicker;

echo DatePicker::widget([
    'language' => 'ru',
    'name'  => 'country',
    'clientOptions' => [
        'dateFormat' => 'yy-mm-dd',
    ],
]);
```

- Aplikasi Yii2 merupakan aplikasi yang menganut konsep *Object Oriented Programming* yang dibangun menggunakan ***Components***
- ***Components*** merupakan sebuah (extended) class/objek dari yii\base\Component.
- Yii application pun sebenarnya juga merupakan sebuah komponen. (Yii application dapat diakses melalui **`\Yii::$app`**)
- Class/objek komponen pada Yii2 adalah objek PHP dengan tambahan tiga fitur utama, yaitu:
  - Properties
  - Events
  - Behaviors
- Salah satu contoh komponen adalah DatePicker widget. (lihat gambar disamping)
  - Komponen ini dapat digunakan dengan mudah di view untuk memilih tanggal secara interaktif.
  - Properti dari komponen juga dapat diubah dengan mudah untuk menyesuaikan konfigurasinya.
- ***Components*** di Yii2 sangat powerful, namun membutuhkan sumber daya (CPU & memory) yang cukup besar.
- Jika class/objek komponen yang anda buat tidak membutuhkan *events* dan *behaviors*, anda dapat membuatnya dengan meng-extend `yii\base\Object` (instead of **`yii\base\Component`**).

Custom Component

- Jika diperlukan, anda dapat membuat custom component sendiri, dengan cara:
  - Buat sebuah file php yang berisi custom component anda dan letakkan di directory components (buat jika belum ada).
  - Tambahkan komponen anda kedalam file konfigurasi (web.php)
- Anda dapat memanggil custom component anda melalui **`\Yii::$app->nama_komponen`**




```
$config = [
  'id' => 'basic',
  'basePath' => dirname(__DIR__),
  'bootstrap' => ['log'],
  'components' => [
    'myComponent' => [
      'class' => 'app\components\MyComponent',
    ],
// ...
```

```
// Inside some view
<h2> Header:
  <?php \Yii::$app->myComponent->welcome() ?>
</h2>
// ...
```

Properties

- Pada PHP, variable di dalam sebuah class (objek) juga disebut properties.
- Yii2 mengenalkan sebuah cara penanganan properties menggunakan method *getter* dan *setter*.
  - Method *getter* adalah method yang namanya dimulai dengan kata *get.* Sedangkan *setter* dimulai dengan kata *set*.
  - Anda dapat mengakses method getter atau *setter* ini seperti mengakses sebuah property biasa, dengan menggunakan kata setelah kata *get* ataupun *set* merupakan (lihat contoh disamping)
- Dengan *getter* dan *setter*, anda dapat membuat sebuah property pada class memiliki "karakteristik" tertentu.
  Misal: anda dapat membuat sebuah string property yang isinya selalu huruf kecil.
- Untuk membuat class (objek) yang memiliki kemampuan ini, anda harus meng-extend class tersebut dari yii\base\Object.

```php
// MyComponent.php
<?php
  namespace app\components;
  use yii\base\component;

  class MyComponent extends Component
  {
    public $label1;
    private $_label2;

    public function getLabel2(){
      return $this->_label2;
    }

    public function setLabel2($value){
      $this->_label2 = strtolower($value);
    }

  }
?>
```

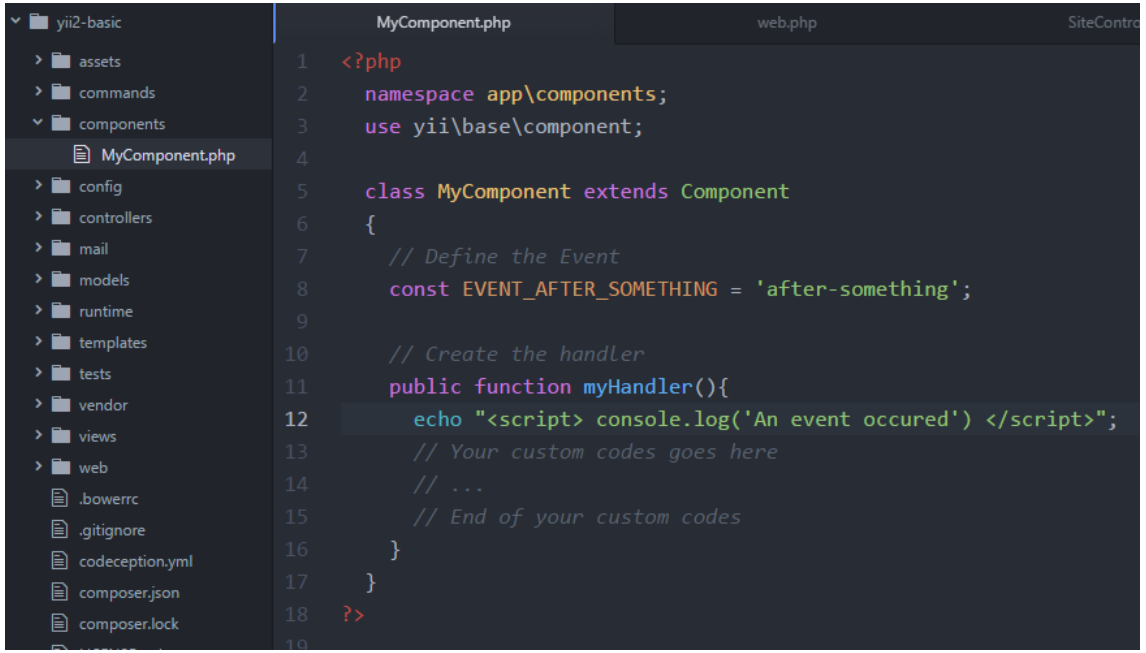
```php
Yii::$app->myComponent->label1 = 'My Label1';
echo "Label1: " . Yii::$app->myComponent->label1;
echo "<br/>";
Yii::$app->myComponent->label2 = 'My Label2';
echo "Label1: " . Yii::$app->myComponent->label2;
```

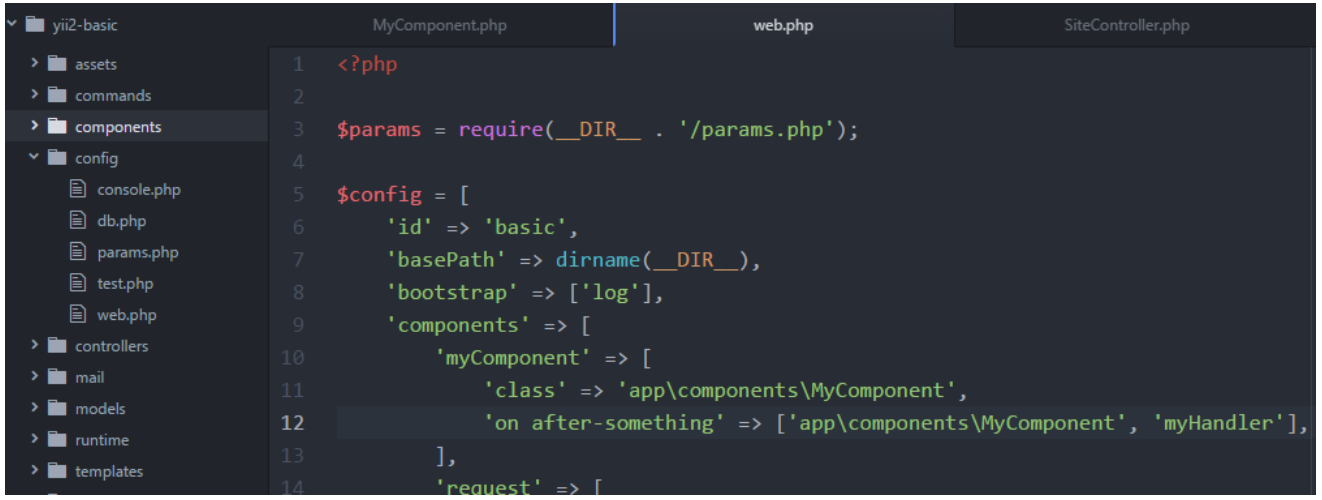
Events

- ***Event*** dapat diartikan sebagai "suatu kejadian".
- Dengan ***Event*** memungkinkan kita memasukkan kode program tertentu kedalam kode program lainnya di sembarang tempat.
- Sebuah ***Event*** dapat dikaitkan (*attach*) dengan satu atau lebih ***Event Handler***.
- ***Event Handler*** adalah method yang akan secara otomatis dijalankan ketika sebuah ***Event*** terjadi (*triggered*).
- Contoh penggunaan Event:
  - Anda ingin menghitung jumlah pengguna yang mengakses route "item/index" dan "item/create"
  - Anda ingin aplikasi mengirimkan email ke admin jika ada pengguna yang menghapus data
  - Anda ingin membuat notifikasi jika ada user baru.
  - Dan lain-lain

Menggunakan Events

1. Definisikan *Event*
   - Definisi event akan lebih baik menggunakan konstanta (const) untuk menghindari kesalahan penulisan
2. Buat sebuah *method* yang berisi hal-hal yang akan dikerjakan jika terjadi suatu *Event*. Method ini disebut sebagai *Event Handler*.
3. Kaitkankan (a*ttach*) *Event* ke *Event Handler*
4. *Trigger the Event* menggunakan method *trigger()*






```php
public function actionAbout()
{
    Yii::$app->myComponent->trigger(\app\components\MyComponent::EVENT_AFTER_SOMETHING);
    return $this->render('about');
}
```

Behaviors

- Dengan *behaviors* anda dapat meningkatkan kemampuan sebuah komponen tanpa harus mengubah komponen tersebut.
- Anda dapat meng-attach *behaviors* pada component dengan tehnik *statically* atau *dynamically*.
- ***Statically***: anda dapat meng-attach *behavior* pada sebuah component dengan meng-override method *behaviors()*.
- ***Dynamically***: anda dapat meng-attach *behavior* pada sebuah component dengan memanggil method *attachBehavior()*.

```
class SiteController extends Controller{
  // Attach behavior statically in Controller
  public function behaviors() {
    return [
      'access' => [
        'class' => AccessControl::className(),
        'only' => ['logout'],
        'rules' => [
          [
            'actions' => ['logout'],
            'allow' => true,
            'roles' => ['@'],
          ],
        ],
      ],
    ];
  }
// ...
```

```
use app\components\MyBehavior;

// attach a behavior object dynamically
$component->attachBehavior('myBehavior1', new
MyBehavior);

// attach a behavior class dynamically
$component->attachBehavior('myBehavior2',
MyBehavior::className());
```

Behaviors (2)

```
class Item extends \yii\db\ActiveRecord
{
 // Attach behavior statically in Model
  public function behaviors()
  {
    return [
    \yii\behaviors\TimestampBehavior::className(),
    \yii\behaviors\BlameableBehavior::className(),
  ];
}
// ...
```

Yii2 menyediakan beberapa built-in behavior, diantaranya:

- **AccessControl** behavior (contoh pada slide sebelumnya): digunakan untuk mem-filter hak akses pengguna terhadap *controller action*.
- **Timestamp** behavior (contoh disamping): digunakan untuk mencatat waktu pembuatan dan perubahan data di table secara otomatis.
  - Anda perlu membuat kolom ***created_at*** dan ***updated_at*** dengan tipe data ***int*** pada table terkait
- **Blameable** behavior (contoh disamping): digunakan untuk mencatat user_id yang melakukan pembuatan ataupun perubahan data di table secara otomatis.
  - Anda perlu membuat kolom ***created_by*** dan ***updated_by*** dengan tipe data ***int*** pada table terkait

Configurations

- *Configurations* banyak digunakan ketika membuat objek baru ataupun menginisialisasi objek yang sudah ada.
- Format dari sebuah *configuration* dapat dideskripsikan dalam bentuk array seperti berikut:

```
[
    'class' => 'ClassName',
    'propertyName' => 'propertyValue',
    'on eventName' => $eventHandler,
    'as behaviorName' => $behaviorConfig,
]
```

- *Configuration* untuk Yii application merupakan salah satu bentuk *configuration* yang sangat komplek.
- Sebagian besar dari widget juga dibentuk menggunakan *configuration*.

```
$config = [
    'id' => 'basic',
    'basePath' => dirname(__DIR__),
    'bootstrap' => ['log'],
    'components' => [
        'mycomponent' => [
            'class' => 'app\components\MyComponent',
        ],
        'request' => [
```

```
]);
echo Nav::widget([
    'options' => ['class' => 'navbar-nav navbar-right'],
    'items' => [
        ['label' => 'Home', 'url' => ['/site/index']],
        ['label' => 'About', 'url' => ['/site/about']],
        ['label' => 'Contact', 'url' => ['/site/contact']],
        Yii::$app->user->isGuest ? (
            ['label' => 'Login', 'url' => ['/site/login']]
        ) : (
            '<li>'
            . Html::beginForm(['/site/logout'], 'post'
                                , ['class' => 'navbar-form'])
            . Html::submitButton(
                'Logout (' . Yii::$app->user->identity->username . ')',
                ['class' => 'btn btn-link']
            )
            . Html::endForm()
            . '</li>'
        )
    ],
]);
NavBar::end();
```

Aliases

- Alias digunakan untuk merepresentasikan path maupun URL.
- Sebuah alias harus dimulai dengan karakter "@"
- Untuk mendefinisikan alias dapat anda lakukan dengan cara:

```
// an alias of a file path
Yii::setAlias('@foo', '/path/to/foo');
// an alias of a URL
Yii::setAlias('@bar', 'http://www.example.com');
```

- Anda juga dapat mendefinisikan alias berdasarkan alias yang lain

```
Yii::setAlias('@foobar', '@foo/bar');
```

Aliases (2)

- Alias juga dapat didefinisikan pada application's configuration (web.php)

```
'aliases' => [
        '@foo' => '/path/to/foo',
        '@bar' => 'http://www.example.com',
],
```

- Untuk menggunakan alias dapat anda lakukan dengan memanggil method Yii::getAlias(). Contoh:

```
echo Yii::getAlias('@foo'); // displays: /path/to/foo
echo Yii::getAlias('@bar'); // displays: http://www.example.com
echo Yii::getAlias('@foo/bar/file.php'); // displays:
/path/to/foo/bar/file.php
```

- Di beberapa tempat, alias dapat digunakan tanpa harus memanggil method Yii::getAlias(). Contoh:

```
$cache = new FileCache([
        'cachePath' => '@runtime/cache',
]);
```

Pre-defined Aliases

- **@yii**, the directory where the **BaseYii.php** file is located (also called the framework directory).
- **@app**, the base path of the currently running application.
- **@runtime**, the runtime path of the currently running application. Defaults to @app/runtime.
- **@webroot**, the web root directory of the currently running web application. It is determined based on the directory containing the entry script.
- **@web**, the base URL of the currently running Web application. It has the same value as yii\web\Request::$baseUrl.
- **@vendor**, the Composer vendor directory. Defaults to @app/vendor.
- **@bower**, the root directory that contains bower packages. Defaults to @vendor/bower.
- **@npm**, the root directory that contains npm packages. Defaults to @vendor/npm.

Class Autoloading

- Aplikasi Yii bergantung pada mekanisme class autoloading untuk mengikutsertakan class-class yang dibutuhkan
- Autoloader dipasang ketika anda meng-include-kan file Yii.php
- Secara default, Yii menggunakan dua class autoloader, yaitu Composer autoloader dan Yii autoloader, yang diletakkan pada entry script (file index.php).
- Jika anda menggunakan beberapa class autoloader, pastikan Yii autoloader ditempatkan paling akhir.

```php
<?php

// comment out the following two lines when deployed to production
defined('YII_DEBUG') or define('YII_DEBUG', true);
defined('YII_ENV') or define('YII_ENV', 'dev');

require(__DIR__ . '/../vendor/autoload.php');
require(__DIR__ . '/../vendor/yiisoft/yii2/Yii.php');

$config = require(__DIR__ . '/../config/web.php');

(new yii\web\Application($config))->run();
```

Latihan

- Gunakan **Event** untuk melakukan pencatatan data statistic.
  Trigger **Event** ini di action index dan view pada frontend application!
  (Langkah ini untuk menggantikan mekanisme yang sama pada latihan sebelumnya)
- Tambahkan Timestamp dan Blameable behavior pada model Item!

15

Terima Kasih